GMI Components: Import/Export Variables

Jules Kouatchou

May 31, 2006

Issues Raised at the Meeting

```
  - Include the dimensions of variables

    Done

  - Include the units of vraiables

    Done whenever available

  - GMAO will provide "press3e" (atmospheric pressure at the edge
    of each grid box) only and not "press3c", "ai", "bi", "am",
    "bm" and "pt". These quantities will have to be derived by GMI.


%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
WE CAN AND WILL PROVIDE PRESS3C. YOU SHOULD ASSUME THIS.
}

    Though "ai", "bi", "am", "bm" and "pt" may be read in from a
    MetFields file, there is a GMI routine that sets their values
    (that remain constant throughout the code). We can then easily
    derive "press3c". As a matter of fact, we do not even need
    "press3e". We only need the surface pressure.

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
WE ARE TRYING TO AVOID THIS, IF POSSIBLE. THIS IS A TOPICS OF
SOME CONTENTION EVEN IN THE GMAO. THE POINT IS THAT BY READING
(OR DEFINING) AI, BI, ETC. GMI IS ASSUMING A VERTICAL COORDINATE
THAT MAY NOT BE CONSISTENT WITH THE GCM'S. AT PRESENT IT IS
CONSISTENT, BUT IF WE CAN AVOID THIS ASSUMPTION, WE MAY AVOID
PROBLEMS IN THE FUTURE.

THE PROPOSED SOLUTION IS NOT TO REDEFINE AI, ETC OR GET THEM FROM
A MET FILE, BUT TO GO WHEREVER THEY ARE USED AND REPLACE
THEM. WITHIN THE GMI CODE, THE AI AND BI SHOULD BE USED ONLY TO
```

COMPUTE THE PRESS3E, WHILE THE AM AND BM ONLY TO COMPUTE THE
PRESS3C. INSTEAD OF DOING THESE COMPUTATIONS JUST REPLACE THE
RESULT WITHTHE APPROPRIATE PRESS3{EC}.

I ASSUME THE SCALER PT WILL NO LONGER BE NEEDED ONCE THIS
REPLACEMENT IS MADE. IF IT IS, USE THE ARRAY PRESS3E(:,:,0)
INSTEAD. SAME FOR SURFACE PRESSURE---USE PRESS3E(:,:,LM).
}

  - The array "cmi_flags" use for lightning parametrization could
    be removed for now as IMPORT variable. It is resolution and
    MetFields dependent.

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
WE WILL SIMPLY PROVIDE CMI AND LWI AS THE SAME THING. CAN WE FIND OUT
WHAT THE DISTINCTION IS? LEAVE THEM AS IS FOR NOW
}

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
WE NEED TO LOOK AT THIS
}

  - Do the variables "con_precip" and "tot_precip" include rainfall
    or snow?

    Do not know!
%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
HOW DO WE FIND OUT?
}
  - How are "max_cloud" and "ran_cloud" used?

    Inside the Dry Deposition, Emission operators, they are used
    to compute the fractional cloud cover (local variable):

```
     cloud_frac(:,:) =  1.0d0 -  &
                 ((1.0d0 - Maxval (max_cloud(:,:,:), dim=3)) *  &
                  (Product ((1.0d0 - ran_cloud(:,:,:)), dim=3)))
```

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
THIS IS THE TOTAL (2D) CLOUDINESS. IF THIS IS THE ONLY USE IN DRY
DEPOSITION, WE WILL GIVE YOU CLOUD_FRAC DIRECTLY. JUST MAKE IT AN

IMPORT AND HAVE EXTERNAL GMI CODE PROVIDE IT WITH THIS CALCULATION.
}

Inside Gas Phase Chemistry, they are used to derived "tau_cloud"
(DAO MetFields only).

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
IS TAU_CLOUD 2D OR 3D. CAN YOU SHOW ME THIS CALCULATION? WHAT DOES DAO
MET FIELDS ONLY MEAN?
}

Inside the photolysis package, they are use to compute the
fractional cloud cover.

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
TOTAL? AS IN DRY DEP.?
}
Inside the AerosolDust module, they are used to compute the
optical depth of aerosol/dust species.

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
DOES THE AEROSOL MODULE USE THESE OPTICAL DPETHS OR ARE THEY JUST
BEING COMPUTED FOR OUTPUT OR TO COMPUTE OTHER OUTPUT QUANTITIES. THIS
IS WHAT I SUSPECT AND IF SO WE DO NOT NEED THEM. WE WILL HOWEVER NEED
A WAY OF COMPUTING THEM IN THE GCM. THIS WILL REQIRE CONSIDERABLE
FURTHER DISCUSSION.
}
- Reduce the dimensions of the emission array, "emiss(:,:,:,:,:)"
and the species concentartion array, "const(:,:,:,:)".

A possible solution has been proposed by Tom.

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
ARE WE PROCEEDING WITH THIS? CAN THE FINAL INTERFACE AFTER THE
SOLUTION IS IMPLEMENTED BE FULLY DESCRIBED IN THE NEXT ROUND OF EMAILS?
}

- Check the units of "mcor" and "mass" and they need to be IMPORT
variables.

"mcor" and "mass" can internally be derived in the GMI code.

The GMI unit for "mass" is [kg] but GMAO wants [kg/m2].
Do not know how to address the issue.

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
THIS IS AEROSOL RELATED ONLY?? IF MCOR AND MASS ARE IN KG, AT SOME
POINT THE GMI CODE MUST WANT THE AREA OF THE GRID BOX AS INPUT. AT
THAT POINT WE CAN SEE HOW TO REPLACE THE UNITS. AGAIN THIS IS AN ITEM
FOR THE CODE WALKTHROUGH.
}

- Why are "emiss_isop", "emiss_monot" and "emiss_nox" considered
  EXPORT variables for the Emission component?

  The three variables are computed by the Emission component and
  passed to the Chemistry component where they are used.

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
ARE THEY COMPUTED OR SIMPLY READ FROMA FILE AND INTERPOLATED IN TIME
AND/OR SPACE? PHYSICAL COMPUTATIONS REMAIN HERE. IF IT IS ONLY
INTERPOLATION THEY CAN BE REMOVED COMPLETELY AND MADE IMPORTS OF
CHEMISTRY WHERE THEY WILL BE FILLED FROM THE FILES BY GEOS_GENERIC.
}

- Is it enough to provide "precipitation" only instead of the variables
  "rain", "rain_zm", "rain_hk" and "rain_ls"?

  They are mainly employed in the wet deposition package.

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
LETS WALK THROPUGH THIS. IT SHOULD BE EASY TO GIVE THESE A MORE
GENERIC INTERPRETATION THAN THE SHANG-MACFARLANE AND HACK DEFINITIONS.
}

- What are the arays "s_radius", "s_velocity" and "diffaer"?

  The are updated by the module computing the gravitational settling
  of aerosols and passed to the Dry Deposition operator. We may ignore
  them for this work (till we have a coupled aerosol/combo mechanism).

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
YOU WOULD THINK WE NEED THIS. WHAT IS A COMBO?
}

```
   - What is the array "tropp" and its unit?

     There is a routine in GMI that computes its value using the temperature
     and the atmospheric pressure (at the center of the grid box).
     The unit of this variable is [mb].

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
I ASSUME THIS IS THE TROPOPAUSE PRESSURE AND IT SHOULD BECOME AN
IMPORT THAT WE  PROVIDE. AGAIN EXTERNAL GMI CODE CAN DO THE
COMPUTATION AND PASS THE SAME IMPORT.
}

   - How are the files read by the Emission and Chemistry components?

     Most of the files are read by each worker processor.
     A couple of them are read by the master processor (see below).

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
LET'S DO A WALKTHROUGH ON THESE READS AND THE FILE FORMATS
}
```

We list here the varaibles GMI will need from GEOS5 and the ones GMI will pass to GEOS5.
I WILL MARK UP ON THIS TABLE THINGS THAT STAY AS IS (!), THINGS THAT ARE IN QUESTION (?) AND THINGS THAT I THINK SHOULD GO (X)

```
=========================================================================
                          EMISSION Component
=========================================================================

IMPORT Variables
----------------

! press3c      (:,:,:)  Atmospheric pressure at the center of each grid box [mb    ]
! press3e      (:,:,:)  Atmospheric pressure at the edge of each grid box   [mb    ]
? zmmu         (:,:,:)  Z-M convective mass flux in     updraft             [Pa/s  ]
! cmi_flags    (:,:)    Array of flags that indicate continental, marine, or ice
                        Only used for lightning parameterization.
! lwi_flags    (:,:)    Array of flags that indicate land, water, or ice
? dtrn         (:,:,:)  Detrainment rate                                    [DAO:kg/m^2*s]
                                                                            [NCAR:s^-1]
! radswg       (:,:)    Net downward shortwave radiation at ground          [W/m^2 ]
! surf_air_temp(:,:)    Surface air temperature                             [degK  ]
! surf_rough   (:,:)    Surface roughness                                   [m     ]
```

```
? con_precip   (:,:)    Convective precipitation                              [mm/day]
? tot_precip   (:,:)    Total precipitation                                   [mm/day]
! ustar        (:,:)    friction velocity                                     [m/s   ]
? max_cloud    (:,:,:)  Maximum overlap cloud fraction for LW
? ran_cloud    (:,:,:)  Random  overlap cloud fraction for LW
! kel          (:,:,:)  Temperature                                           [degK  ]
! pbl          (:,:)    Boundary layer height                                [m     ]
! humidity     (:,:,:)  Specific humidity                                     [g/kg  ]
? pctm1        (:,:)    Surface pressure at t1                               [mb    ]

? emiss    (:,:,:,:,:)  Array of emissions                                    [kg/s  ]


%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
CONST SHOULD BE INTERNAL AND EXPORT AND ARRAY OF 3D POINTERS
}
x const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]

x pt                    pressure = (am * pt) + (bm * psx)                    [mb   ]
x ai           (:)      Pressure = (ai * pt) + (bi * psx), ai at zone interface
x bi           (:)      Pressure = (ai * pt) + (bi * psx), bi at zone interface

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
HOW ARE LAT-LON USED? WE SHOULD REPLACE THEM
}

X latdeg       (:)      Latitude                                              [deg   ]
X londeg       (:)      Longitude                                             [deg   ]
? mcor         (:,:)    Area of grid box                                      [m^2   ]
? mass         (:,:,:)  Total mass of the atmosphere within each grid box     [kg    ]

EXPORT Variables
----------------

! flashrate    (:,:)    Flash rate (flashes per 4x5 box per s)
! lightning_no (:,:,:)  3d array of pnox production in kg./sec
? emiss_isop   (:,:)    Isoprene    emissions                                 [kg/s  ]
? emiss_monot  (:,:)    Monoterpene emissions                                 [kg/s  ]
? emiss_nox    (:,:)    NOx         emissions                                 [kg/s  ]
? emiss    (:,:,:,:,:)  Array of emissions                                    [kg/s  ]
? const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]


========================================================================
                        WET DEPOSITION Component
========================================================================

IMPORT Variables
----------------

! press3c      (:,:,:)  Atmospheric pressure at the center of each grid box [mb    ]
! press3e      (:,:,:)  Atmospheric pressure at the edge of each grid box   [mb    ]
! kel          (:,:,:)  Temperature                                         [degK  ]
```

```
? rain         (:,:,:)  rainfall across cell edges                          [mm/day]
? rain_zm      (:,:,:)  rain production due to deep conv processes       [kg/kg/sec]
? rain_hk      (:,:,:)  rain production due to shal conv processes       [kg/kg/sec]
? rain_ls      (:,:,:)  rain production due to lasrge-scale processes    [kg/kg/sec]
? coscen       (:)      cosine of latitude of zone centers = cos(dlatr)
! grid_height  (:,:,:)  height of each grid box                           [m     ]
! mass         (:,:,:)  total mass of the atmosphere within each grid box   [kg    ]
! wet_depos    (:,:,:)  wet deposition accumulated since last output        [kg/m^2]
? mcor         (:,:)    Area of grid box                                  [m^2   ]
? con_precip   (:,:)    Convective precipitation                          [mm/day]
? tot_precip   (:,:)    Total precipitation                               [mm/day]
? const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]

EXPORT Variables
----------------
! wet_depos    (:,:,:)  wet deposition accumulated since last output        [kg/m^2]
! const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]
? moistq       (:,:,:)  moisture changes due to wet processes             [g/kg/day]


======================================================================
                        DRY DEPOSITION Component
======================================================================

IMPORT Variables
----------------

X pt                    pressure = (am * pt) + (bm * psx)                   [mb   ]
X ai           (:)      Pressure = (ai * pt) + (bi * psx), ai at zone interface
X bi           (:)      Pressure = (ai * pt) + (bi * psx), bi at zone interface
X am           (:)      pressure = (am * pt) + (bm * psf), am at zone midpoint
X bm           (:)      pressure = (am * pt) + (bm * psf), bm at zone midpoint
X latdeg       (:)      Latitude                                          [deg   ]
X londeg       (:)      Longitude                                         [deg   ]
X mcor         (:,:)    Area of grid box                                  [m^2   ]
X mass         (:,:,:)  Total mass of the atmosphere within each grid box   [kg    ]

! lwi_flags    (:,:)    array of flags that indicate land, water, or ice
! himudity     (:,:,:)  specific humidity                                 [g/kg  ]
X max_cloud    (:,:,:)  Maximum overlap cloud fraction for LW
X ran_cloud    (:,:,:)  random  overlap cloud fraction for LW
! radswg       (:,:)    net downward shortwave radiation at ground        [W/m^2 ]
! surf_air_temp(:,:)    surface air temperature                           [degK  ]
! surf_rough   (:,:)    surface roughness                                 [m     ]
! ustar        (:,:)    friction velocity                                 [m/s   ]
! psf          (:,:)    surface pressure field at t1, known at zone centers [mb    ]
! kel          (:,:,:)  temperature                                       [degK  ]

? s_radius     (:,:,:)  aerosol radius at bottom layer                    [m     ]
? s_velocity   (:,:,:)  aerosol settling velocity at bottom layer         [m/s   ]
? diffaer      (:,:,:)  aerosol diffusivity at bottom layer               [m^2/s ]

? dry_depos    (:,:,:)  dry deposition accumulated since last output        [kg/m^2]
X const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]

EXPORT Variables
----------------
```

```
! dry_depos    (:,:,:)  dry deposition accumulated since last output        [kg/m^2]
! const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]


=========================================================================
                        SIMPLE DEPOSITION Component
=========================================================================

IMPORT Variables
----------------

! press3c      (:,:,:)  atmospheric pressure at the center of each grid box [mb    ]
X const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]

EXPORT Variables
----------------
! const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]


=========================================================================
                        CHEMISTRY Component
=========================================================================

IMPORT Variables
----------------

X pt                    pressure = (am * pt) + (bm * psx)                    [mb   ]
X ai           (:)      Pressure = (ai * pt) + (bi * psx), ai at zone interface
X bi           (:)      Pressure = (ai * pt) + (bi * psx), bi at zone interface
X am           (:)      pressure = (am * pt) + (bm * psf), am at zone midpoint
X bm           (:)      pressure = (am * pt) + (bm * psf), bm at zone midpoint
X latdeg       (:)      Latitude                                             [deg   ]
X londeg       (:)      Longitude                                            [deg   ]
X dlatr        (:)      latitude of zone center in latitude direction        [rad   ]
X mcor         (:,:)    Area of grid box                                     [m^2   ]
X mass         (:,:,:)  Total mass of the atmosphere within each grid box    [kg    ]
! grid_height           height of each grid box                              [m     ]
!
! press3c      (:,:,:)  Atmospheric pressure at the center of each grid box [mb    ]
! press3e      (:,:,:)  Atmospheric pressure at the edge of each grid box   [mb    ]
! tropp        (:,:)    tropopause pressure                                 [mb    ]
? max_cloud    (:,:,:)  Maximum overlap cloud fraction for LW
? ran_cloud    (:,:,:)  Random  overlap cloud fraction for LW
? tau_cloud    (:,:,:)  optical depth (dimensionless)
! kel          (:,:,:)  Temperature                                         [degK  ]
! humidity     (:,:,:)  Specific humidity                                   [g/kg  ]
? pctm2        (:,:)    Surface pressure at t1+tdt                          [mb    ]
? surf_alb_uv  (:,:)    bulk surface albedo (fraction 0-1)
! cmf          (:,:,:)  convective mass flux                              [kg/m^2*s]

! emiss_isop   (:,:)    Isoprene    emissions                               [kg/s  ]
! emiss_monot  (:,:)    Monoterpene emissions                               [kg/s  ]
! emiss_nox    (:,:)    NOx         emissions                               [kg/s  ]
! emiss    (:,:,:,:,:)  Array of emissions                                  [kg/s  ]

X const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]
```

```
EXPORT Variables
----------------

! const      (:,:,:,:)  Species concentration, known at zone centers  [mixing ratio]
? emiss    (:,:,:,:,:)  Array of emissions                                 [kg/s  ]




%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{\it
WE WILL NEED TO KNOW ALL OF THESE FORMATS AND DISCUSS A SANE
CONSISTENT FORMAT (HDF) FOR ALL OF THEM.
}




=========================================================================
                  INPUT FILES NAMES
=========================================================================

    Emission
emiss_infile_name     : emission                input file name (netCDF)
                              read by all the worker processors
light_infile_name     : lightning               input file name (ascii)
                              read by all the worker processors
precip_infile_name    : precipitation           input file name (ascii)
                              read by all the worker processors
soil_infile_name      : soil type               input file name (ascii)
                              read by all the worker processors
isopconv_infile_name  : isoprene convert        input file name (ascii)
                              read by all the worker processors
monotconv_infile_name : monoterpene convert     input file name (ascii)
                              read by all the worker processors
veg_infile_name       : vegetation type         input file name (ascii)
                              read by all the worker processors
lai_infile_name       : leaf area index         input file name (ascii)
                              read by all the worker processors
gcr_infile_name       : Galactic Cosmic Ray     input file name (netCDF)
                              read by all the worker processors
fertscal_infile_name  : fertilizer scale        input file name (ascii)
                              read by all the worker processors

    Chemistry
forc_bc_infile_name   : forcing bc              input file name (ascii)
                              read by master processor
h2oclim_infile_name   : water climatology       input file name (netCDF)
                              read by all the worker processors
lbssad_infile_name    : liq bin sul sad         input file name (netCDF)
                              read by all the worker processors

uvalbedo_infile_name  : uv albedo               input file name (ascii)
                              read by master processor
```

```
cross_section_file    : X-Section quantum yield input file name (ascii)
                                read by all the worker processors
rate_file             : Master rate             input file name (ascii)
                                read by all the worker processors
T_O3_climatology_file : T & O3 climatology      input file name (ascii)
                                read by all the worker processors

=========================================================================
      s_radius, s_velocity and diffaer
=========================================================================
s_radius, s_velocity and diffaer are updated inside the routine
computing the gravitational settling of aerosols, Update_Grav_Settling.
Update_Grav_Settling is called only if the logical variable "do_grav_set"
is set to true.
```

How some Variables are Used in the GMI Code

```
=========================================================================
    mcor, pt, ai, bi, am, bm
=========================================================================

==================
EMISSION Component
==================
"mcor" is employed to
       - update the array "emiss_nox" (unit issue?)
       - update diagnostics variables

"humidity, ai, bi, pt" are only used to compute the grid box height
inside Add_Emiss_Llnl. They can be removed from the argument list of
the emission control routine and be replaced by "grid_height".

==================
CHEMISTRY Component
==================
"mcor" is utilized
       - to calculate column ozone in the lookup table module (unit issue?)
       - for surface emission diagnostics.

"mcor, ai, bi" are used to calculate conversion to go from kg/box/s to
mole/cm^3/s.

"humidity, ai, bi, pt" are only used to compute the grid box height
inside Update_Semiss_Inchem. They can be removed from the argument list of
this routine and be replaced by "grid_height".
The same can be said for the argument list of
     - Update_QuadChem
     - Update_Smv2chem

"ai, bi, pt" are employed to compute tau_cloud for DAO Met Fields.

"ai, bi, am, bm, pt" are used in the photolysis package to calculate
pressure at boundaries of CTM levels:
```

```
   Press(:) = ai(:)*pt + bi(:)*SurfPressure
   Press(:) = am(:)*pt + bm(:)*SurfPressure

====================
DEPOSITION Component
====================
"mcor" is utilized to
       - update dry_depos and wet_depos (unit issue?)
       - update moistq and precip_bot (unit issue?)

"humidity, ai, bi, pt" are only used to compute the grid box height
inside Update_Drydep. They can be removed from the argument list of
this routine and be replaced by "grid_height".
"humidity, am, bm, pt" are also used there to compute something
similar to the grid box height.

====================
CONVECTION Component
====================
"mcor" is utilized to
       - compute the internal number of time step for convection
       - update the wet_depos array (unit issue?)

"ai, bi, pt" only employed in Do_Convec_Dao2.

====================
ADVECTION Component
====================
"mcor, ai, bi, pt" are used the total mass for internal diagnostics.
The variable "mass" can passed in the argument list of the advection
control routine and replace them.
```

From 4D variables to 3D variables

```
! Module implements a derived type to support an "array of arrays" in F90.
! The derived type is not opaque - a compromise to minimize the impact on
! legacy code that uses the original data structure (a 4D array).

module GMI_Bundle_mod
   implicit none
   private

   public :: GMI_Bundle_type ! derived type
   public :: GMI_Bundle      ! constructor
   public :: clean           ! destructor
   public :: setPointer      ! accessor
   public :: getPointer      ! accessor

   ! Derived type has public components for ease of use.
   ! Strong encapsulation would require intrusive changes
   ! in bothe GEOS and GMI.
   type GMI_Bundle_type
      real (kind=r64) :: pArray3D(:,:,:) => null()
   end type GMI_Bundle_type
```

```
   interface clean
      module procedure cleanScalar
      module procedure cleanVector
   end interface

contains

   ! Initialize a bundle from an already allocated 4D array.
   function GMI_Bundle(concentrations) result (bundle)
      real (kind=r64), target :: concentrations(:,:,:,:) ! I,J,K,species
      type (GMI_Bundle_type), pointer :: bundle(:)

      integer :: i
      integer :: numSpecies

      numSpecies = size(concentrations, 4)

      allocate(bundle(numSpecies))
      do i = 1, numSpecies
         call setPointer(bundle(i), concentrations(:,:,:,i))
      end do

   end function GMI_Bundle

   ! Establish a link to a specified 3D target
   ! (pretend that encapsulation is enforced)
   subroutine setPointer(this, newTarget)
      type (GMI_Bundle_type) :: this
      real (kind=r64), target :: newTarget(:,:,:)

      ! The following could fail on some compilers if a temporary copy of
      ! concentrations is generated.  Unlikely in this context, though.
      this%pArray3D => newTarget

   end subroutine setPointer

   ! Retrieve pointer (pretend that encapsulation is enforced)
   function getPointer(this) result(ptr)
      type (GMI_Bundle_type) :: this
      real (kind=r64), pointer :: ptr

      ptr => this%pArray3D

   end function getPointer

   ! Nullify the internal pointer (pretend encapsulation is enforced)
   subroutine cleanScalar(this)
      type (GMI_Bundle_type) :: this
      nullify(this%pArray3D)
   end subroutine cleanScalar

   ! Clean an array of derived types. First clean each element, and
   ! then deallocate the array.
   subroutine cleanVector(this)
      type (GMI_Bundle_type), pointer :: this(:)
      integer :: i
```

```
      do i = 1, size(this)
         call clean(this(i))
      end do
      deallocate(this)

   end subroutine cleanVector

end module GMI_bundle_mod
```